Pandavas and Kauravas are ruling two different kingdoms sepertaed by river.

If you connect both the kingdoms they will be in Square shaped land. Land

occupied by the kingdoms, indiacted with 1's and the river, indiacted with 0's.

Now, Pandavas and Kauravas decided to build a bridge on the river for easy

connectivity. As the cost of building a bridge will be high, they have decided

to reduce the length of the bridge. You are allowed to build the bridge on

the river cells, connected by 4 directions only (up, down,left,right).

Your task is to help the Kings, to minimize the occupation of river cells, to

build the bridge with minimum cost. And return the count of river cells occupied.

Input Format:

-------------

Line-1: An integer N, size of the land.

Next N lines: N space separated integers, either 0 or 1.

Output Format:

--------------

Print an integer result.

Sample Input-1:

---------------

4

1 1 1 0

1 0 0 0

1 0 0 1

0 0 1 1

Sample Output-1:

----------------

2

Sample Input-2:

---------------

5

1 1 0 0 0

1 1 0 0 0

0 0 0 0 1

0 0 0 1 1

0 0 1 1 1

Sample Output-2:

----------------

3

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int[][] arr=new int[n][n];

boolean[][] vis=new boolean[n][n];

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

arr[i][j]=sc.nextInt();

}

}

Queue<int[]> q=new LinkedList<>();

boolean flag=false;

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

if(arr[i][j]==1){

flag=true;

dfs(arr,i,j,vis,q);

}

if(flag){

break;

}

}

if(flag){

break;

}

}

System.out.println(bfs(arr,q,vis)-1);

}

public static int bfs(int[][] arr,Queue<int[]> q,boolean[][] vis){

while(!q.isEmpty()){

int[] curr=q.poll();

int r=curr[0];

int c=curr[1];

int count=curr[2];

if(arr[r][c]==1){

return count;

}

if(r-1>=0 && !vis[r-1][c] ){

q.add(new int[]{r-1,c,count+1});

vis[r-1][c]=true;

}

if(r+1<arr.length && !vis[r+1][c]){

q.add(new int[]{r+1,c,count+1});

vis[r+1][c]=true;

}

if(c-1>=0 && !vis[r][c-1] ){

q.add(new int[]{r,c-1,count+1});

vis[r][c-1]=true;

}

if(c+1<arr.length && !vis[r][c+1] ){

q.add(new int[] {r,c+1,count+1});

vis[r][c+1]=true;

}

}

return -1;

}

public static void dfs(int[][] arr,int r,int c,boolean[][] vis,Queue<int[]> q){

if(r<0 || r>=arr.length || c<0 || c>=arr.length || arr[r][c]==0|| vis[r][c]==true){

return;

}

vis[r][c]=true;

arr[r][c]=0;

q.add(new int[]{r,c,0});

dfs(arr,r-1,c,vis,q);

dfs(arr,r+1,c,vis,q);

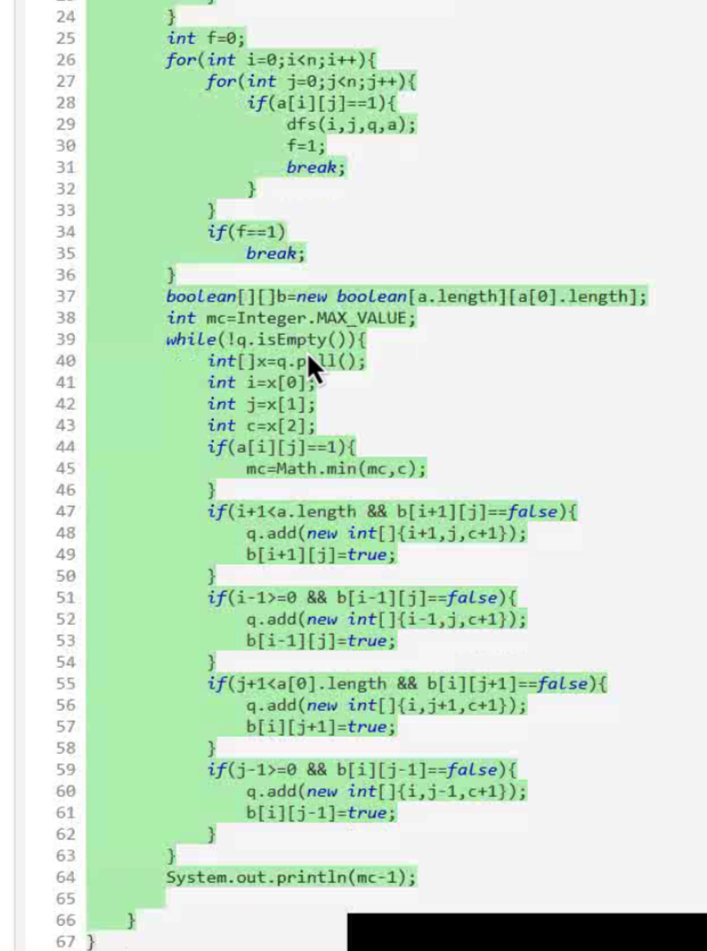
dfs(arr,r,c-1,vis,q);

dfs(arr,r,c+1,vis,q);

}

}





Shakuni playing 'Paramapadasopanam' game with Dharamaraja.

Shakuni wants to win everytime. He has a boon that whenever he used to

roll the dice, the dice roll to show his desired number.

Rules

--------

The game is played with a cubic dice of 6 faces numbered 1 to 6.

And Paramapadasopanam of size N\*N.

- Starting from square X = 1, reach the square N\*N with some rolls of the dice

- Once Shakuni roll the dice, he will land on one of the followin squares only

X+1, X+2, X+3, X+4, X+5, X+6.

- If current value at position X is not equal to -1, then there is a ladder

or snake.

- If the position of X is at the base of a ladder, the player must climb

the ladder. Ladders go up only.

- If the position of X is at the mouth of a snake, the player must go down

the snake and come out through the tail. Snakes go down only.

You have to findout, what would be the least number of rolls required

for shakuni to win.

Note:

Shakuni can only take a snake or ladder at most once per move:

if the tail of a snake or ladder is the start of another snake or ladder,

you do not continue moving.

Input Format:

-------------

Line-1: An integer n , size of Paramapadasopanam.

Next N lines: N space separated integers, either -1 or

an integer between 1 to n\*n.

Output Format:

--------------

Print an integer, least number of rolls required for shakuni to win.

Sample Input-1:

---------------

6

-1 -1 -1 -1 -1 -1

-1 -1 -1 -1 -1 -1

-1 -1 -1 -1 -1 -1

-1 35 -1 -1 13 -1

-1 -1 -1 -1 -1 -1

-1 15 -1 -1 -1 -1

Sample Output-1:

----------------

4

Explanation:

------------

At the beginning, you start at square 1.

Roll the dice to land at square 2, and must take the ladder to square 15.

Roll the dice to land at square 17, and must take the snake to square 13.

Roll the dice to land at square 14, and must take the ladder to square 35.

Roll the dice to land at square 36, Shakuni Won.

Total number of rolls are 4.

